Problem 17: Conway's Game of Life

Points: 40

Author: Louis Ronat, Denver, Colorado, United States

Problem Background

In 1940, computer scientist John von Neumann defined life as a creation which can reproduce itself and simulate a Turing machine: briefly, a device which acts according to a set of rules. This definition gave rise to a continuing series of mathematical experiments. Among the most famous of these is a "game" created by mathematician John Conway in 1970 called *Life*. Conway's *Life* consists of a set of four rules to be followed by a computer given an initial state of a grid filled with "live" and "dead" cells.

In each generation:

1. Any live cell adjacent to one or zero live cells dies (from loneliness).
2. Any live cell adjacent to two or three live cells lives.
3. Any live cell adjacent to four or more live cells dies (from overcrowding).
4. Any dead cell adjacent to exactly three live cells becomes alive (through reproduction).

Diagonal cells are considered to be adjacent. *Life* evolves by applying these rules to the "world" represented by the grid. The rules are applied, the world is redrawn, the rules are applied again, and the world is redrawn again, repeating indefinitely

T = 0 **1** **2** **3** **4**

Page 2

These seemingly simple rules are completely deterministic; that is, each generation is determined entirely by the state of the previous generation. Despite this, these rules can yield some very complex behavior. Theoretically, *Life* is a "universal Turing machine;" this means that anything that can be calculated through an algorithm can be calculated with *Life*.

Problem Description

You must design a program that implements Conway's *Life* on a 10-by-10 grid. Your program will be given an initial state for the first generation. It must then determine the state of the world after a given number of generations have been performed. Note that cells outside the bounds of the 10-by-10 grid are always considered dead.

Sample Input

The first line of your program's input, **received from the standard input channel**, will contain a positive integer representing the number of test cases. Each test case will include the following lines of input:

- A line containing a positive integer, **X**, indicating the number of generations to

calculate

- Ten lines containing ten characters each representing the initial state of the world. Characters will be either '1', representing a "live" cell, or '0', representing a "dead" cell.

1
6
0000000000
0000000000
0000000000
0000010000
0000111000
0000111000
0000010000
0000000000
0000000000
0000000000

Sample Output

For each test case, your program must output the state of the world after the indicated number of generations. Each test case should include ten lines with ten characters each.

Page 3

0000000000
0000000000
0000111000
0001000100
0000000000
0000000000
0001000100
0000111000
0000000000

0000000000